Transport Protocols and Applications for Internet Use in Space[1]

Ed Criscuolo, Keith Hogie, Ron Parise
Computer Sciences Corp
7700 Hubble Dr.
Lanham-Seabrook MD 20706
(301) 794-4072 Ed.Criscuolo@gsfc.nasa.gov

Abstract—An Internet datagram delivery service between space systems only provides end-to-end addressability. Building systems and performing actual spacecraft operations requires a variety of services operating over the Internet datagram delivery service. This paper discusses ways to use the capabilities of the upper layer Internet protocols to support the varied communication needs of satellites. It focuses on protocols in the transport layer (layer 4) and application layer (layer 7) which use the basic packet delivery capabilities of the Internet Protocol (IP) and the network layer (layer3).

The transport layer primarily adds data stream multiplexing and reliable data delivery options for use by applications. Data stream multiplexing is provided by the port mechanism in the User Datagram Protocol (UDP) and the Transport Control Protocol (TCP). UDP provides a basic packet delivery service similar to that used in today's spacecraft while TCP provides an automatic retransmission capability for reliable data stream delivery. Data streaming is also supported by the Real Time Protocol (RTP) which operates over UDP. Each of these protocols has benefits and limitations in various space communication environments with a range of link errors, propagation delays, and bit rates. Transport protocol selection and operational usage are discussed with respect to satellite communication requirements.

Finally, actual spacecraft operations are performed by using applications running over transport protocols. The use of standard Internet applications such as NTP, FTP, SMTP, and telnet is discussed with respect to satellite operational requirements. The actual use and performance of many of these protocols by the Operating Missions as Nodes on the Internet (OMNI) project at NASA/GSFC on orbit with the UoSAT-12 spacecraft is also described.

TABLE OF CONTENTS

1. INTRODUCTION

This paper will discuss the use of standard Internet applications and protocols to meet the technology challenge of providing dynamic communication among heterogeneous instruments, spacecraft, ground stations, and constellations of spacecraft. The objective is to characterize typical mission functions and automated end-to-end transport of data in a dynamic multi-spacecraft environment using off-the-shelf, low-cost, commodity-level standard applications and protocols. These functions and capabilities will become increasingly significant in the years to come as both Earth and space science missions fly more and more sensors and the present labor-intensive, mission-specific techniques for processing and routing data become prohibitively expensive. This work is about defining an architecture that allows science missions to be deployed "faster, better, and cheaper" by using the technologies that have been extremely successful in today's Internet.

2. OVERVIEW OF INTERNET PROTOCOLS IN SPACE

The goal of the OMNI project is to define and demonstrate an end-to-end communication architecture for future space missions. The authors have combined their knowledge and experience in Internet technologies and space communication systems in developing the following end-to-end data communication concept.

End-to-End Network Concept

The data communication requirements of many advanced space missions involve seamless, transparent connectivity between space-based instruments, investigators, ground-based instruments and other spacecraft. The key to an architecture that can satisfy these requirements is the use of applications and protocols that run on top of the Internet Protocol[1] (IP). IP is the technology that drives the public Internet and therefore draws billions of dollars annually in research and development funds. Most private networks also utilize IP as their underlying protocol. IP provides a basic standardized mechanism for end-to-end communication between applications across a network. The protocol provides for automated routing of data through any number of intermediate network nodes without affecting the endpoints.

Proposed Architecture

Recognizing the clear benefits of IP as an end-to-end networking protocol, the OMNI project developed a reference system architecture for the space and ground

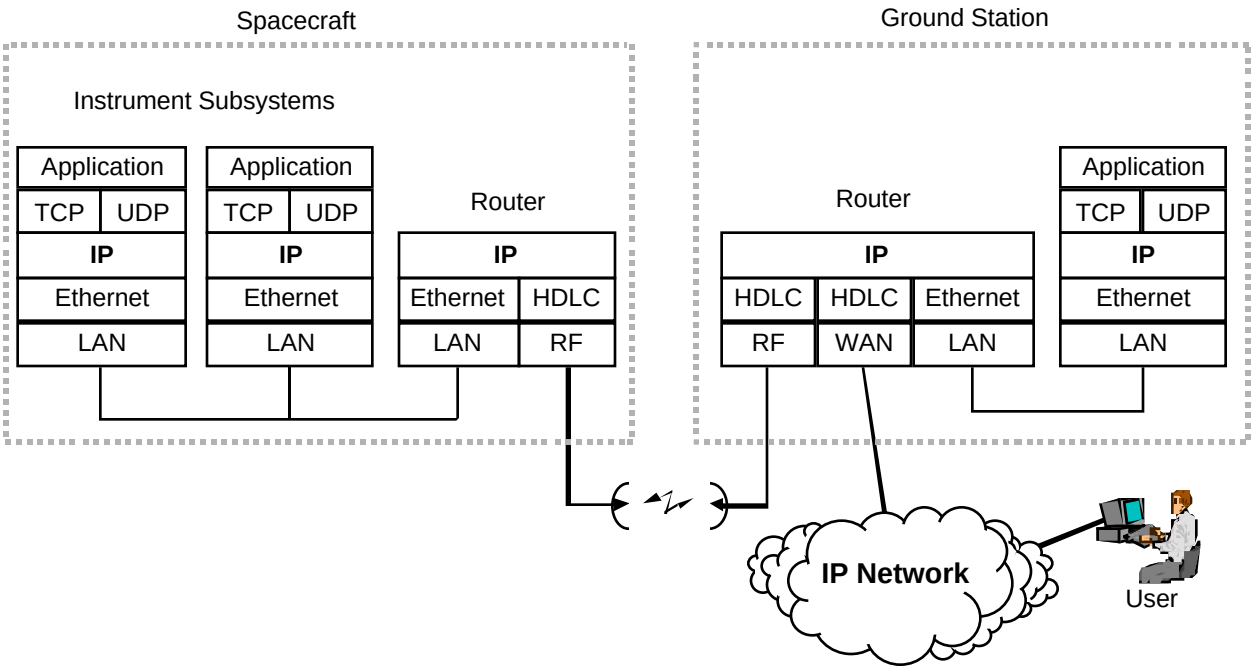

Figure 1 – System Architecture for an IP Mission

segments of future IP missions. The goals were to maximize the use of commercial-off-the-shelf (COTS) hardware and protocols while avoiding creating any new "space-specific" solutions. A high-level view of this architecture appears in figure 1. Several notable features are present, and are discussed in the following sections.

3. Layered Model

The Internet suite of protocols, and the OMNI reference architecture, is based on the OSI seven-layer model of networking, but with some differences. In the OSI model, there are seven distinct layers. Starting from the lowest, they are:

1. Physical - Raw bits, coding (wire, fiber, RF)
2. Link - Packets (HDLC,FDDI, ATM, ethernet)
3. Network - end-to-end addressed datagrams (IP)
4. Transport - multiplexed packets (TCP, UDP)
5. Session - login, authentication
6. Presentation - formating, translation
7. Application - user data

In the Internet implementation, layers 5 - 7 tend to be compressed into a single application layer. For example, the Internet file transfer application "FTP" incorporates elements of the session layer (user login), presentation layer (translation of ASCII files), and application layer (transfer user files).

The important thing to note in the system architecture in figure 1 is the commonality of IP as the network layer. This allows everything above the network layer to operate independently of the type of link used. Similarly, in the lower layers, different hardware drivers can be substituted without affecting the specific applications being run.

4. Lower Layers

In the reference architecture developed by the OMNI project, certain choices were made for the Physical Layer (RF & coding), link layer (HDLC) and network layer (IP). The issues and rationale behind these choices are beyond the scope of this paper. These important topics are the subject of a separate paper titled "*Link and Routing Issues for Internet Protocols in Space*".[2]

5. Transport Layer

The transport layer has the major function of providing stream (or packet) multiplexing of multiple channels into a single link. This logical multiplexing is distinct from any physical-resource multiplexing that occurs lower down in the link layer. All Internet transport layer protocols provide this capability, commonly referred to as "ports" or "sockets".

This multiplexing capability is one of the most important aspects of Internet transport layer protocols. It provides the ability to transparently mix thousands of unrelated asynchronous data streams on a single physical link, without the applications that generate the data needing to be aware of each other or the layers below. This "virtual channel" capability even provides for intermixing multiple transport protocols on the same physical link. Each transport protocol has its own separate set of 65,535 ports.

Prioritization of these "virtual channels" is handled down in the network layer (layer 3) by assigning different queuing priorities to unique port-protocol combinations. This is a standard feature found in most IP routers in use today, and was successfully used by the OMNI project to prioritize mixed streams of data.

These capabilities match well with the telemetry requirements of modern spacecraft, which often have hundreds of "application IDs", representing many separate asynchronous data streams running at different priorities.

Beyond multiplexing, Internet transport protocols have a wide range of different capabilities and limitations. There are two main transport protocols currently in wide use on the Internet: Transport Control Protocol[3] (TCP) and User Datagram Protocol[4] (UDP). Although TCP is the most well known of these protocols, it is important to make the distinction that not all IP is TCP/IP. In addition, a third transport protocol, Realtime Transport Protocol[5] (RTP), is in common use but is generally implemented "on top of" UDP instead of as a separate protocol with its own protocol id. Each protocol has it's own strengths and weaknesses.

Selection of a transport protocol for a particular type of spacecraft or instrument data is mission specific, and would be driven by the mission requirements and system engineering tradeoffs. There is no single "one size fits all" answer.

UDP

UDP is a connectionless transport protocol designed to operate over IP. Its primary functions are error detection and multiplexing. UDP does not guarantee the delivery or order of packets, but guarantees that if a packet is ever delivered with errors, such errors will be detected. Because the UDP format is simple, it has a low overhead. See figure 2. It is also fast compared to TCP since there is no connection establishment phase.

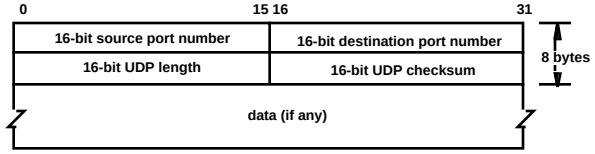

Figure 2 - UDP Header Layout

UDP provides "atomic packet delivery". This means that the application will never see a partial or fragmented packet (regardless of any fragmentation and reassembly performed by the lower layers). Delivery of a packet to the application layer is all-or-nothing.

These characteristics make UDP the protocol to use when the timeliness of the data is more important than getting every packet. Examples of this include spacecraft engineering data, health & safety telemetry, and blind commanding.

UDP is a "send-and-forget" protocol. Packets are addressed to their network endpoint and sent on their way without any connection phase or handshaking. This has both advantages and disadvantages. On the plus side, it means that the protocol will work with highly asymmetric or unidirectional links, is completely delay-insensitive, and supports multi-cast. These characteristics make it well suited for deep-space missions, such as Mars. On the minus side, UDP does not provide flow control or reliable transport. If these features are required with UDP, they must be built on top of it at the application layer, as in today's spacecraft protocols. The following section on the Application Layer discusses several examples of UDP-based applications that take this approach.

RTP

RTP is used to carry data that has real-time properties. It provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. Those services include payload type identification, sequence numbering, timestamping, and delivery monitoring. Applications typically run RTP on top of UDP to make use of its multiplexing and checksum services, but both protocols contribute parts of the transport protocol functionality. See figure 3 for a diagram of the additional 12 bytes of header that RTP adds onto UDP. RTP (and UDP) supports data transfer to multiple destinations using multicast distribution if provided by the underlying IP network.

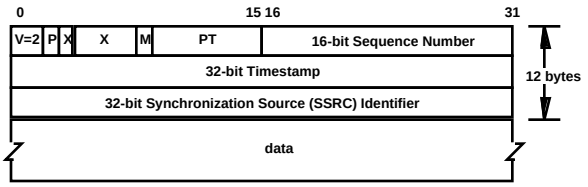

Figure 3 - RTP Header Layout

RTP is most well known for streaming audio and video on the web, in products such as Real-Video and QuickTime, and is used for Voice-over-IP (VoIP). Although originally designed for supporting audio and video over packet networks, RTP is also useful for transporting *any* isochronous data where the timing of the data is important.

RTP provides hooks for adding in reliability and flow control if these features are required.

TCP

TCP is a connection oriented transport protocol designed to work in conjunction with IP. TCP provides the application layer with the ability to *reliably* transmit a byte stream to a destination, and allows for multiplexing multiple TCP connections on a single host. It provides flow control, and has "out-of-band" handling for priority messages. A diagram of the TCP header is shown in figure 4.

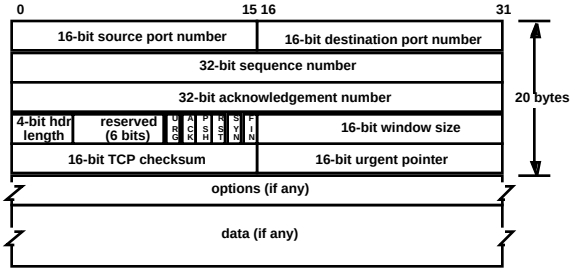

Figure 4 - TCP Header Layout

Being connection oriented, TCP requires a connection setup phase, followed by a data transmission phase. A connection is terminated when it is no longer in use.

The reliability and flow control of TCP requires that status information be sent with each packet, and acknowledgement be received back from the recipient. This allows TCP to recover from data that is damaged, lost, duplicated, or sent out of order.

These characteristics make TCP a protocol to use when the overriding requirement is for the error-free transfer of data.

Examples of this include downloading instrument science data, and uploading spacecraft or instrument command loads. Many off-the-shelf applications are built on top of TCP and can perform this function. In many cases, reliable transfer of instrument data files directly to the scientist can *completely replace* traditional level-0 processing.

Along with TCP's capabilities come some limitations.

Because of the handshaking and flow control, TCP requires a bi-directional link. This link can exhibit only a moderate amount of asymmetry (~50:1) before throughput is affected.

Because TCP is a windowed, buffered protocol, it is sensitive to the Round Trip Time (RTT) delay. With increasing RTT, larger window buffers are required in order to maintain throughput. But larger buffers exact a larger penalty when a packet is lost and has to be retransmitted. In practical terms, what this means is that TCP will perform fine out to about lunar distance. In particular, TCP has been successfully used at geosyncronous distance at over 400 megabits per second[6].

TCP currently has no mechanism for distinguishing loss due to congestion from loss due to noise. This means that increasing noise will reduce throughput as retransmission timeouts increase. The OMNI project has performed experiments which have shown that single-session TCP bandwidth utilization falls to under 60% at an error rate of 10^{-5}; however, through the use of forward error correction codes, (FEC), most space missions operate at an error rate of 10^{-7} or better. At these rates, TCP's bandwith utilization approaches its theoretical maximum of ~92%. In addition, current standards activities, such as Explicit Congestion Notification[7] (ECN), Selective Acknowledgement[8] (SACK), and TCP/Peach[9], are underway to make TCP more "satellite friendly".

6. Application Layer

While transport protocols enable the exchange of data, more functionality is required to perform useful work. This is the domain of the application layer.

There are many standard application layer protocols, such as HTTP, SMTP, FTP and Telnet. Most of these are defined in RFCs, are widely implemented, and interoperate universally. Many of them are capable of handling a large variety of space mission requirements. Self-defined protocols are also possible. Designing your own protocol for your application has the advantage of being flexible, lightweight, and efficient, but has the disadvantage of being non-interoperable with other applications. Each mission needs to make that decision based on its own requirements and the cost/benefit tradeoffs.

UDP based Applications

UDP applications can be divided up into several categories that are relevant to spacecraft operations.

Simple data delivery — In general, a simple custom application would be required to wrap application-specific telemetry or command packets in a self-defined protocol. This is functionally equivalent to current space missions using CCSDS framing and packet protocols. It may even make sense to use the CCSDS packet (not frame) formats inside of UDP, as the CCSDS packet structure is already defined.

Reliable file transfer with UDP — A number of standard applications/protocols are available to perform reliable file transfer via UDP. These include Pacsat Broadcast Protocol[10] (PBP), Multicast File Transfer Protocol[11] (MFTP), CCSDS File Delivery Protocol[12] (CFDP), Network File System[13] (NFS), and Trivial File Transfer Protocol[14] (TFTP). PBP, MFTP and CFDP are of particular interest for deep space missions because they can operate over a mostly unidirectional link. They send out all the file's packets nonstop without waiting for any handshake. Sometime later, (a minute, an hour, a day) a brief return link is required to transmit a list of NAKs for any packets that were lost. These packets then get resent on the next contact. This makes these protocols delay insensitive, just the thing needed for missions at L1 or Mars and beyond.

Time Synchronization — The Network Time Protocol[15] (NTP) is a UDP based protocol and application that is used to synchronize the time of a computer client or server to another server or reference time source. Typical NTP configurations utilize multiple redundant servers and diverse network paths, in order to achieve high accuracy and reliability. Some configurations include cryptographic authentication to prevent accidental or malicious protocol attacks. A space mission using NTP would typically place its primary timeserver right at the groundstation to minimize delay variations and maximize security.

TCP Based Applications

TCP applications can be divided up into several categories that are relevant to spacecraft operations.

Reliable Simple Data Delivery — As with UDP, a simple custom application would be required to wrap application-specific telemetry or command packets in a self-defined protocol. The difference here is that TCP takes care of automatically performing any retransmissions required to guarantee delivery of every data byte. In the commanding case, this is similar to current missions using the CCSDS COP-1 protocol. In the telemetry case, there is no current system implemented to perform automatic retransmissions and reliably deliver every data byte. Current missions either tolerate any lost data or manually retransmit the entire data set a second time in an attempt to fill in any losses.

Reliable File Transfer with TCP — There are two main application-level TCP-based file transfer protocols that are widely available and instantly familiar to anyone who uses the Internet: File Transfer Protocol[16] (FTP) and Hypertext Transfer Protocol[17] (HTTP). FTP is the older of the two, and tends to be more efficient in its use of persistent connections, but is more complex to implement. HTTP, on the other hand, is extremely simple to implement in a small memory footprint, but sets up and tears down a connection for every transfer.

Hundreds of implementations of each of these protocols are available both commercially and as freeware applications. These are a very good match for the needs of science missions, which often have large quantities of prerecorded data that must get shipped to the scientist with maximum fidelity.

Email — Simple Mail Transfer Protocol[18] (SMTP), as defined in STD-10/RFC-821, specifies the protocol used to send electronic mail (e-mail) between TCP/IP hosts. E-mail is probably the most widely used TCP/IP application. The basic Internet mail protocol provides mail and message exchange between TCP/IP hosts. Facilities have been added for the transmission of binary data which cannot be represented as 7-bit ASCII text.

SMTP is based on end-to-end delivery; an SMTP client will contact the destination host's SMTP server directly to deliver the mail. It will keep the mail item being transmitted until it has been successfully copied to the recipient's SMTP server. SMTP servers can also be set up as "mail gateways" to implement a "store and forward" delivery system. In either case, the mail is always addressed to the end user.

In space-based applications, SMTP can provide the scientists and spacecraft operators the capability of sending and receiving commands and data when they are not in contact with the spacecraft, and having those files automatically queued and delivered without further human intervention. This "batch" or "bundled" mode of data transfer very closely matches the requirements of many space missions. It can be easily and cost-effectively accomplished with commercial off-the-shelf applications without inventing any new "space-specific" protocols.

7. SPACE VS TERRESTRIAL ISSUES

There are a number of apparent issues for space-based usage of Internet protocols. These are often misunderstood or misrepresented. A recent quote in Space News stated: "*The environment for the Internet is basically no delays, no errors, continuous connectivity, and pretty symmetric data transfer. If you look at the space environment, it's almost completely reversed. There are high delays and high error rates. The links are not continuous or symmetric.*" If this description of the Internet were true, we would all have continuous 100 MBit/Second connectivity to our PCs and cell phones. Instead, we have 56 kBit/Second dialup modems, micropower cell phones that run 2400 baud *if* they can make a connection, and network delays that sometimes run up into the seconds. So, even though on the surface it would appear that "space is special" and has unique problems, upon careful examination, each of these problems can either be found to be a non-problem, or to have a terrestrial parallel that has been solved in the commercial world.

Long Delay

Often it is stated that space missions *must* be carried out with "Round trip delays much greater than ground systems"[19], and that "...long propagation times cause terrestrial protocols to operate sluggishly or fail outright"[19]. For low earth orbit (LEO) missions, which represent the *large majority* of space missions, this is simply not true. A LEO spacecraft is only 200-400 miles away when it passes overhead. Since RF travels at the speed of light, this translates into only a 4 mS round trip time! Even at the horizon, which for a spacecraft in a 400 mile high orbit is approximately 3000 miles away, this is about a 32 mS round trip time. Compare this with typical Internet ping times from Baltimore to Los Angeles of 100 mS and the LEO spacecraft should actually run TCP/IP *better* than coast-to-coast terrestrial links. Even out to geosynchronous orbit, the round trip delay time is only 240 mS. Experiments have been performed at the NASA Glenn Research Center[6] using the ACTS satellite, which have operated TCP at over 400 megabits/second at this distance. These experiments used ACTS as a "bent pipe", so a round trip required two hops to geosynchronous distance, or around 480 mS. TCP is limited by its bandwidth-delay product, requiring a transmission window buffer of equal or greater size. This means that low-bandwidth/high-delay TCP connections are similar to high-bandwidth/low-delay ones. Laboratory experiments have suggested that lunar distance, with its 1666 mS round trip time, would require some care in setting up the connection, and represents the practical limit for TCP based applications. Beyond this distance, deep space missions, such as Mars, should look to using one of the delay-insensitive UDP based protocols, such as MFTP, PBP, or CFDP.

Noise

Frequently, it is pointed out that most packet losses on the Internet are due to congestion, whereas most losses on a space-to-ground link are due to noise. TCP has no mechanism for distinguishing packet loss due to noise from packet loss due to congestion, so it always assumes congestion and responds to noise by slowing down. This feature of TCP is often used to imply that *all* Internet protocols operate sluggishly or fail outright in the presence of noise. *This is not true for UDP based protocols*. UDP does not perform flow control and never attempts to throttle the data.

Many terrestrial environments feature noisy channels that successfully carry TCP traffic. The best example of this is the ordinary dialup telephone line. The telephone line has a bit error rate (BER) that is similar to most spacecraft RF links. The CCITT recommendations for voice circuits that have been conditioned to carry data[20] is a BER of 10^{-5}. Similarly, NASA typically specs its spacecraft RF links at a BER of 10^{-5}. The reason TCP works over the phone lines is that the modem applies error correction down at the physical layer, transparently to the upper layers. This allows the upper layers to behave as though they have a clean link. Similarly, NASA applies error correction to it's space links, achieving operational BERs down to 10^{-7} or better. At 10^{-7}, handshaking protocols, such as TCP/IP, work well.

In addition to this, current standards work is in progress to make TCP itself less sensitive to uncorrected noise loss. These include ECN, SACK (which is already widely distributed), and TCP/PEACH. These efforts are driven, in part, by the explosive demand for Internet-enabled cell phones and wireless devices, which must operate in an inherently noisy, low power environment.

Power, CPU, and Bandwidth Constraints

The previously mentioned wireless/cell-phone industry must operate in an environment that is far more severely constrained than that of most spacecraft. Electrical power, CPU processing power, and RF bandwidth are limited to an embedded device that fits in a shirt pocket. Much as in a deep space mission "every bit is precious", so ongoing research and development is being aimed at protocols that are efficient and error tolerant, such as IP header compression[21] and Cellular-IP[22]. These efforts are being coordinated through the Internet Engineering Task Force (IETF), so the resulting non-proprietary standards will interoperate and be available to all. In fact, given the huge potential size of the Internet cell phone market, it seems possible that in a few years, a large amount of Internet traffic will flow over cellular protocols.

Intermittent Connectivity and Variable Routing

Spacecraft that are not in a geosynchronous orbit cannot maintain continuous direct contact with the ground. Contacts are limited to a brief time when the spacecraft passes within line-of-sight of the ground station. For a low earth orbit, this "pass" is typically no more than 8 to 15

minutes long, a few times a day. If more contacts are needed, more ground stations must be used, complicating the routing of data to and from the spacecraft.

This situation is very similar to people with laptop computers. They, and their computers, change locations and intermittently connect to the network at different points. But they want to maintain just one IP address. The Mobile–IP[23] protocol was designed to handle just this problem. Using it, mobile users can maintain a single Internet address while connecting to the network at different locations. Through the actions of a "home agent" and one or more "foreign agents", a "care of" address is established that allows transparent end-to-end addressing of data to and from the mobile host. This protocol does exactly what an IP spacecraft needs in order to send and receive data using multiple ground stations.

Forward/Return Path Asymmetry

Most spacecraft have a much greater downlink bandwidth than uplink bandwidth. This asymmetry is often incorrectly attributed to the fact that spacecraft are limited by their power and weight budgets, and cannot generally support large steerable high-gain antennas. While this fact is true, it is not what limits the uplink data rate. Up to a point, any shortcomings of the spacecraft antenna or receiver can be compensated for by more power and bigger antennas on the ground. The real limitation is driven in part by physics, but mostly by convention.

In the early years of space exploration, most missions had modest uplink requirements for commanding. As a result, the standard RF systems for the evolving Spacecraft Tracking and Data Network (STDN) came to modulate the uplink signal on a 16 KHz subcarrier, reserving the main carrier for ranging tones. Although this choice was adequate for the times, today this legacy of "STDN compatibility" limits the uplink channel to about 8 KBits/second instead of the 2 MBits/second that is possible when BPSK modulating the main S band carrier.

In any event, TCP will typically run with asymmetries of up to 50:1 before throughput begins to be affected. For an 8 kbps uplink this corresponds to a 400 kbps downlink. Although far below the maximum possible, *this data rate is adequate for more than half of the current science missions.* And this asymmetry limitation *only* affects TCP. UDP based protocols can always downlink at the full rate.

Missions that want to use TCP above the 400kbps rate will have to use a communication system that is not based on the 16 KHz subcarrier uplink. The NASA Tracking and Data Relay Satellite System (TDRSS) is one such system. Experiments with a ground-based IP spacecraft simulator were able to establish a symmetric 1 MBit/second duplex link through TDRSS using nothing more than a 5 watt transmitter and an 18" steerable patch antenna.[24]

8. IP-Based Operations Scenarios

An IP-based communication architecture can support all existing operations concepts and makes some new, complex concepts realistic.

For example, real-time engineering and housekeeping data can be monitored during a pass using UDP/IP packets. This only requires a uni-directional downlink. On the other hand, if a bidirectional link is available, guaranteed reliable delivery of data packets can be achieved by using TCP/IP. In both cases, only a simple application layer function needs to be written for the flight software. The TCP/IP stack is available as a standard COTS product for current flight operating systems such as VxWorks.

Recorded science and engineering data can be stored onboard in files in a standard COTS file system. These files can later be transferred to the ground with guaranteed complete, time-ordered records using an off-the-shelf application such as FTP. If the round-trip delay times are too great to use a TCP-based protocol such as FTP, a UDP-based protocol, such as Starburst/MFTP, could be used instead.

Onboard clock synchronization, typically a thorny problem, can be handled by using the NTP. NTP can automatically calculate propagation delay times, time variance, and drift rates, relative to one or more reference timeservers. It can set the spacecraft's clock and even perform periodic drift mitigation. The NTP protocol is capable of precision on the order of 240 picoseconds ***if*** sufficient bandwidth and CPU speed are available.

Store-and-forward commanding, and data delivery, can be achieved by using SMTP to deliver the files as email attachments. For example, in the commanding case, the control center emails the command load to the spacecraft as an attachment. A mail server at the groundstation stores the file until the next contact and then automatically transfers it to the spacecraft for processing. Similarly, in the data delivery case, the C&DH processor, or even a "smart" instrument, emails the data file as an attachment to a message sent to the principal investigator. The onboard mail server stores the file until the next contact and then transfers it to the ground for automatic delivery to the owner of the data.

The IP suite supports various commanding scenarios. When the downlink is not available for acknowledgement, "blind" real-time commands can be sent to the spacecraft using UDP. This is required for emergency situations such as rescuing a spacecraft from tumbling. For nominal operations, reliable commanding can be achieved by using TCP, which automatically takes care of performing the handshaking and any necessary retransmissions.

These basic capabilities, and the end-to-end network addressing capability of IP, can be used to support new, complex scenarios. These include user interaction, spacecraft cross-support, and ad-hoc collaborations.

These scenarios also highlight the capabilities needed for constellations of spacecraft. Formation flyers could send messages back and forth to keep their group navigation within specifications. Constellations of nanosats could message their data to larger members with the power for delivery to the ground.

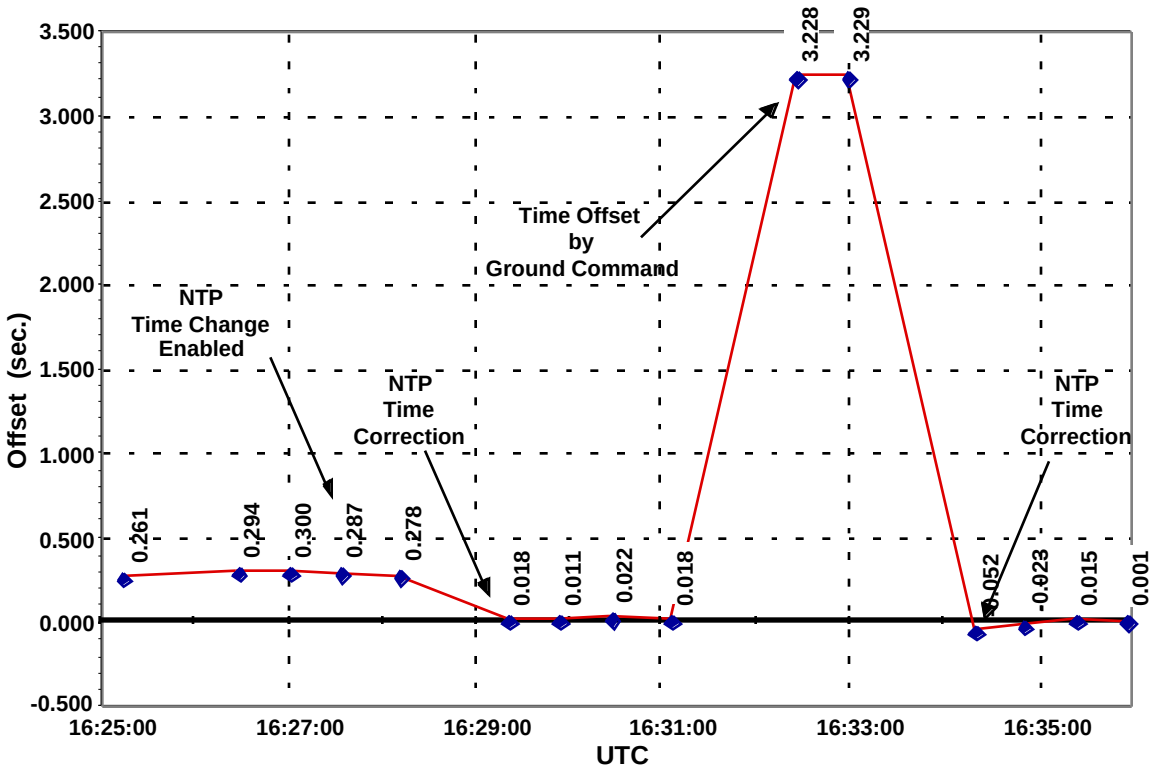

Figure 5 - NTP Clock Synchronization Test Results

9. Present Results

At the present time, the OMNI project has successfuly performed both ground-based and on-orbit validation tests of many of the concepts described in this paper. In particular, on-orbit testing of UDP telemetry delivery, NTP (operating over UDP) and FTP (operating over TCP) were successfully completed. Some results are presented here, but for full details, refer to the papers titled *"Demonstrations of Internet Protocols in Space Using TDRSS"*[24] and *"Results of 'Internet in Space' Tests Using UoSat-12"* [25].

On-Orbit Clock Synchronization with NTP

For the clock synchronization tests, a standard NTP client was ported to the UoSAT-12 spacecraft. It was used to automatically synchronize the onboard clock to UTC. On the ground, the US Naval Observatory's timeserver (tick.usno.navy.mil) was used as the reference timeserver. This represents somewhat of a worst-case test, as the USNO is a quarter of the way around the world, over 20 router hops, from the UoSat-12 ground station in Surrey, UK. In a real operations environment, a timeserver of the required accuracy would be located at the groundstation to minimize the network latency and variation that NTP has to factor out. However, NTP is designed to deal with these factors, and the resulting levels of accuracy might be quite adequate for many space missions even under worst case conditions.

Two tests were performed, both following the same scenario. The tests started out with the onboard NTP server running, but disabled from actually changing the spacecraft's clock. The onboard server periodically negotiated with the USNO timeserver to factor out network delay. If it was successful, the onboard server calculated the offset it thought it had to apply to the spacecraft's clock. This value was sent to the ground in a UDP telemetry stream, where it was logged for later analysis. For purposes of this testing, the NTP negotiation period was set artificially low to 30 seconds so that a reasonable number of data points could be collected during the 14 minute pass. A short time into the test, a command was sent to the spacecraft to enable NTP to actually change the onboard clock. NTP requires two successful offset calculations before it will adjust the clock. Later in the test, a command was sent to the spacecraft to manually set the onboard clock in error by a large amount (2-3 seconds). After two successful offset calculations, NTP should again reset the clock. If the time is off by more than one second, the spacecraft NTP client adjusts to the proper second during one adjustment period, and adjusts for fractional seconds on the next adjustment period.

The results for the test run on April 14, 2000 are shown in figure 5. The pass began with a spacecraft clock offset from UTC of approximately +300 ms. Two calculations after NTP time-changing was enabled, the calculated offset dropped to less than two clock ticks (20 ms) and stayed there until it was manually set in error from the ground. A ground command was used to set the clock ahead by approximately 3.25 seconds. Two offset calculations after that, NTP had reset the clock to within six clock ticks of UTC, taking an additional two offset calculations to settle within two clock ticks of UTC.

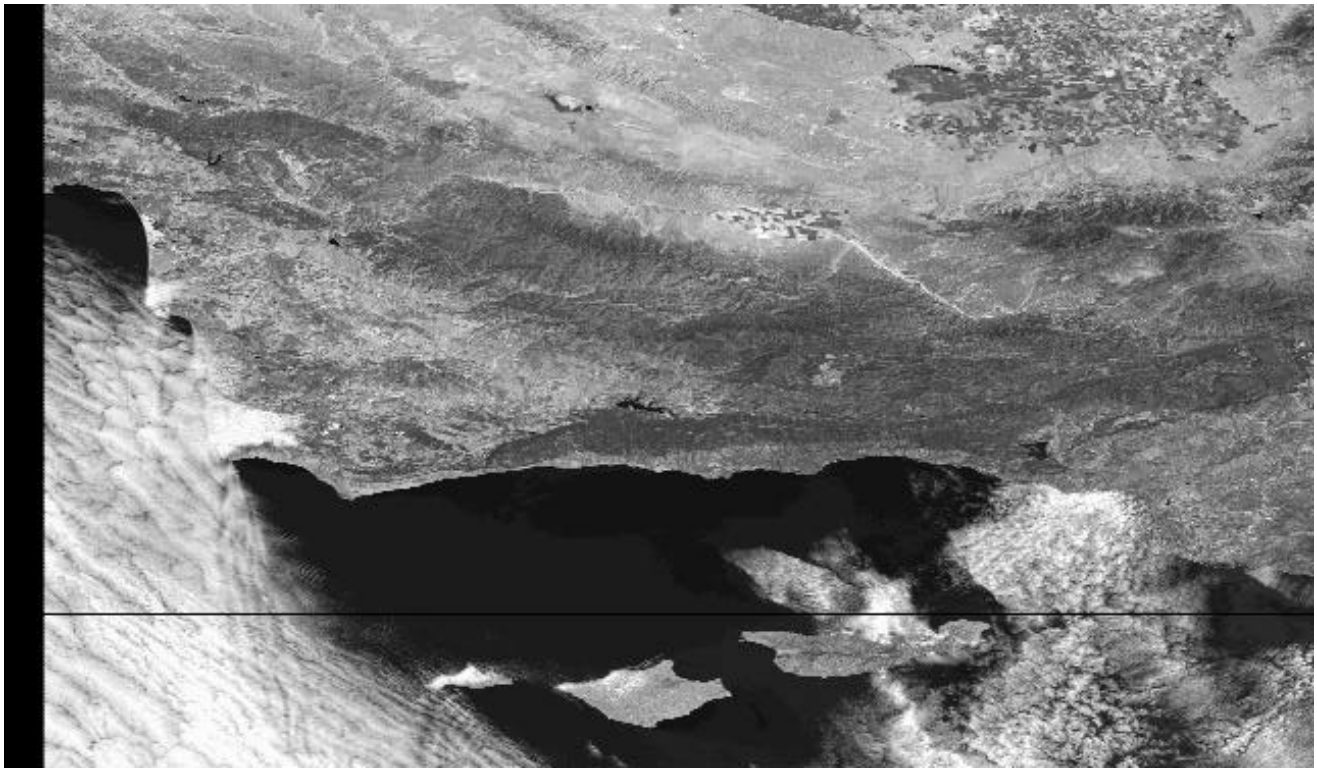

Figure 6 - Landsat-7 Image Showing Noise Induced Scan Line Loss

Error-Free Downloads with FTP

Current space missions, such as Landsat-7, download their image data "open-loop", without any automatic retransmission. Any data lost due to noise is lost forever. As a result, Landsat-7 employs large ground antennas and strong Reed-Solomon forward-error-correction in order to reduce its nominal bit error rate down to the range of 10^{-7} to 10^{-9}. Even so, image data is sometimes still lost. Figure 6 shows a typical "dropped scan line" image that results from a small data loss.

UoSat-12 does not employ *any* forward-error-correction, and its groundstation employs modest antennas. As a result, UoSat-12's downlink only has a typical bit-error-rate between 10^{-5} and 10^{-6}. Conventional open-loop space mission protocols would provide badly degraded data at best. What is required is an application built on top of a protocol that performs automatic retransmissions, such as FTP over TCP.

For the initial FTP tests, a standard FTP server was ported to the UoSat-12 spacecraft. It was used to provide reliable, error-free transport of UoSat-12 image data to both the groundstation and remote user sites using off-the-shelf FTP client applications. For subsequent tests, packet trace software was installed on UoSat-12 and at the ground sites in order to capture and quantify the number and types of lost packets and retransmissions.

Figure 7 shows a result of tests performed on June 7, 2000. This mosaic consists of four sequential images of Perth, Australia that were downloaded from UoSat-12 via FTP with 100% data integrity, despite numerous packet losses. Note the lack of any data loss artifacts, such as dropped scan lines, shear misalignment, or pixelation.

The cost of this reliability in the face of adverse bit-error-rates is a reduction in performance. Our laboratory tests have shown that FTP/TCP performance is not significantly affected at bit-error-rates below 10^{-7}. Above this, performance falls to around 60% bandwidth utilization at 10^{-5}.

Subsequent FTP testing on July 5, 2000 began to characterize the on-orbit performance. Figure 8 shows the packet trace for a typical file download. This 227 kByte file required 445 packets and experienced 9 retransmissions, all due to packet losses on the downlink. These retransmissions are noted by the "O" label, signifying receipt of an out-of-order packet. Seven of the nine were single packet losses, which allowed the TCP "rapid-retransmission" algorithm, but two were multiple packet losses, which resulted in a retransmission timeout. The overall result of these retransmissions was a reduction in bandwith utilization to 79.2% compared to a theoretical maximum of 91.6%. These results are preliminary, but are in good agreement with the laboratory testing.

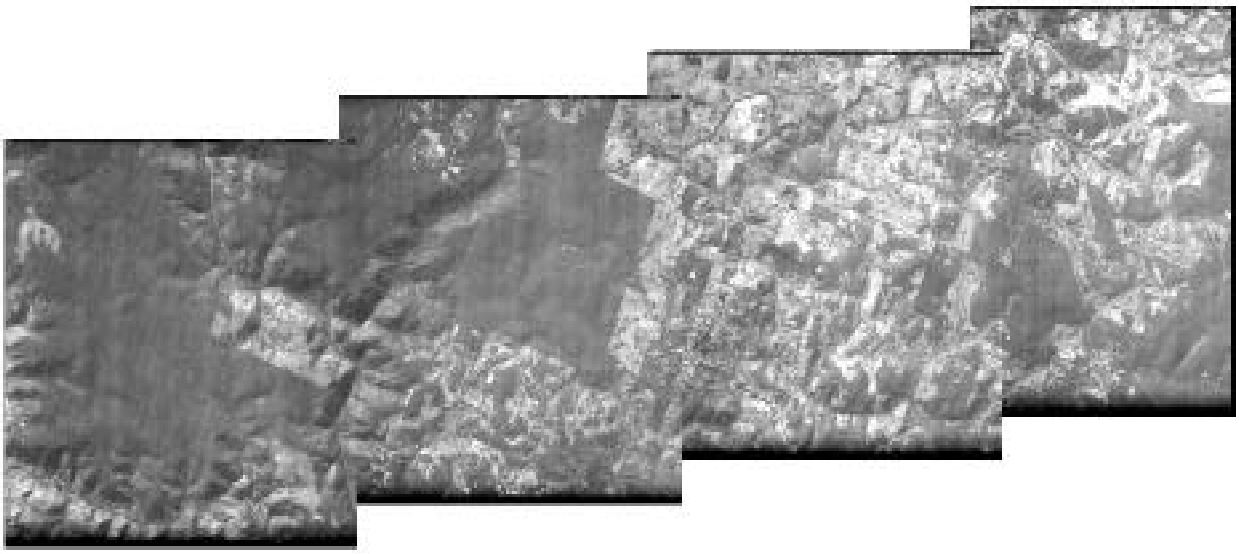

Figure 7 - UoSat-12 Images Downloaded Error-Free with FTP

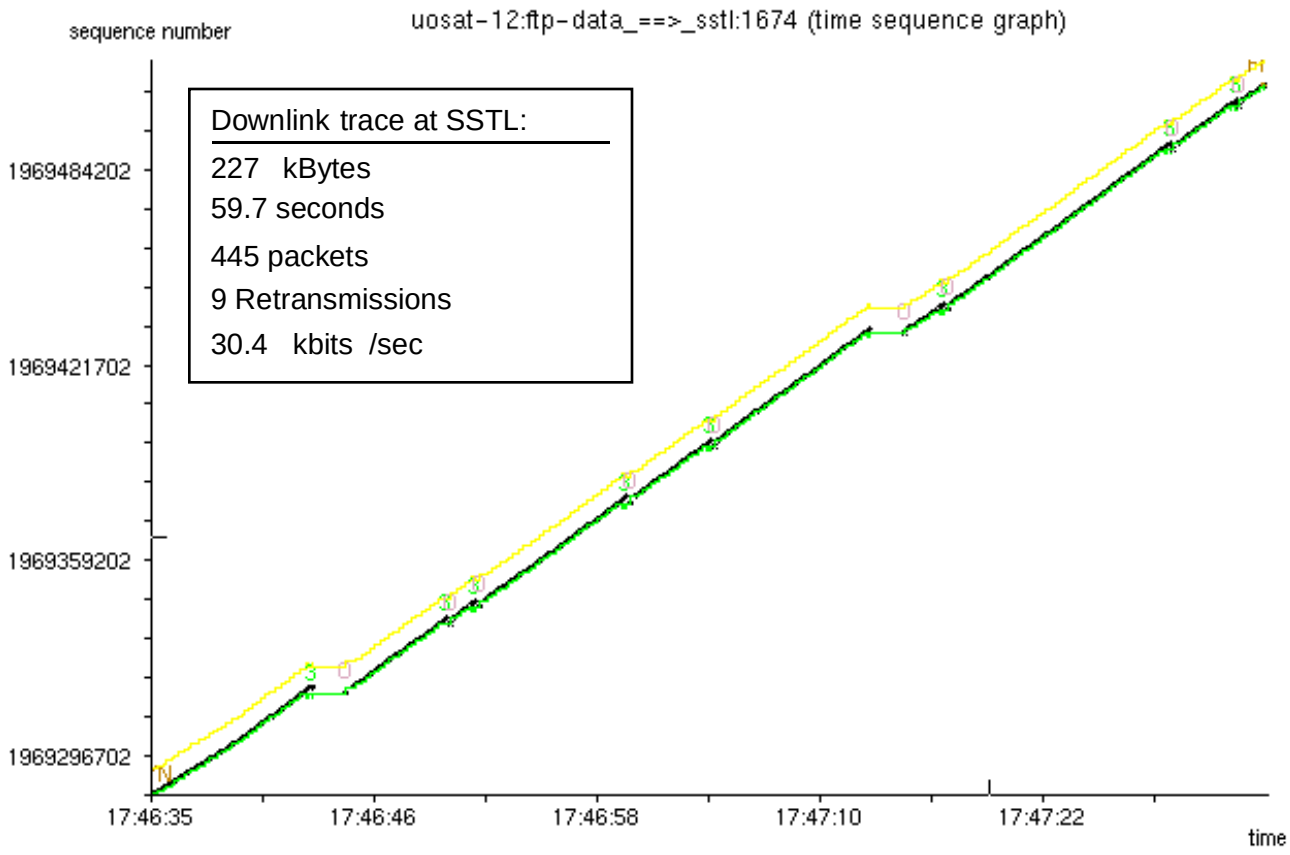

Figure 8 - Packet Trace of UoSat-12 Download with FTP

10. Future Work

The activities so far have been done in fairly simple and controlled configurations. More work is needed to investigate additional protocols required to properly deploy Internet protocols in worldwide, operational space communication networks. Implementing HDLC framing and IP packets on spacecraft and installing routers at ground stations are not major problems. The main issues are to deal with network security and the highly mobile aspects of spacecraft.

The OMNI project is in the process of expanding its test environment to include multiple spacecraft simulators and ground nodes for testing mobile IP and mobile routing

protocols. These investigations plan to use the Linux and VxWorks operating systems on the spacecraft simulators and Cisco IOS 12.1 or newer software on the ground routers.

Security solutions based on Internet security protocols[26] (IPsec) and virtual private networks[27] (VPNs) will be configured and tested along with the mobile IP environment.

The mobile IP and security work will focus on issues for deploying IP in operational space communication networks. Additional work is also planned to identify spacecraft control and data delivery applications to use over a space IP network. One of the main application areas to be investigated will be reliable file transfer in space environments. This will focus on file transfer applications that operate over UDP and that can then operate in communication environments with extremely long round-trip times and link bandwidth asymmetry.

Current information on test results and future activities will be posted on the OMNI project web site at http://ipinspace.gsfc.nasa.gov/.

11. Conclusions

The current activities have demonstrated that standard Internet protocols will function in a space environment and are useful and effective in typical spacecraft operations.

The speed and ease with which UoSat-12 was adapted to use Internet protocols have demonstrated that this is a cost-effective solution for space missions. The conversion of the spacecraft, conversion of the groundstation, and the initial tests were completed in only 5 months at a cost of only $50k.

The last 22 months of tests and demonstrations have shown that HDLC framing and IP packets provide a very simple and flexible communication mechanism for space communication. HDLC framing is well supported in a wide range of COTS products and has been used on over 20 spacecraft for over 10 years. Using the Internet Protocol as a network layer allowed easy integration and testing of our end-to-end scenarios. Also, both HDLC and IP required no modifications to operate in intermittent space link conditions.

While many of the Internet protocols (i.e. TCP, FTP, NTP) work in full-duplex communication scenarios, we have also successfully used others (i.e. UDP) in either receive-only or transmit only scenarios. During the NTP tests described in this paper, a one-way UDP based telemetry stream was used for diagnostics and statistics data. These one-way data transmission modes must be supported in order to deal with spacecraft contingencies when a full-duplex link is not available. This is just one more case of the Internet protocols being flexible enough to support a wide range of requirements.

Finally, introducing a network protocol like IP in the communication architecture has allowed us to easily support a wide range of communication scenarios and mission scenarios. Using IP has allowed us to communicate around the world and introduce new applications very quickly and easily. Most of the traditional interface control documents (ICDs) are eliminated since the Internet standards are already well specified, highly interoperable, and widely available.

12. Acknowledgments

The research described in this paper was carried out by personnel from Computer Sciences Corporation working for NASA's Goddard Space Flight Center under contract GS-35F-4381G S-36130-G, with additional efforts and support contributed by individuals from various GSFC organizations. The work was funded by NASA's Space Operations Management Office (SOMO) Communication Technology Project headed by Tom Costello. The authors would like to thank Dave Israel and GSFC code 450 for their pioneering IP work on the SPTR project, Chris Jackson of Surrey Space Technologies Ltd. and Harold Price of VyTek Wireless for their support on UoSAT-12, and Cisco Systems for the loan of a Cisco 1601 router for use in the SSTL ground station.

REFERENCES

[1] Internet Engineering Task Force, "Internet Protocol, DARPA Internet Program Protocol Specification", RFC–791, September 1981

[2] K Hogie, E Criscuolo, R Parise, "Link and Routing Issues for Internet Protocols in Space", 2001 IEEE Aerospace Conference, March 2001

[3] Internet Engineering Task Force, "Transmission Control Protocol, DARPA Internet Program Protocol Specification", RFC-793, September 1981

[4] Internet Engineering Task Force, "User Datagram Protocol, DARPA Internet Program Protocol Specification", RFC-768, August 1980

[5] Internet Engineering Task Force, "RTP: A Transport Protocol for Real-Time Applications", RFC–1889, January 1996

[6] A Welch, D Brooks, D Beering, D Hoder, M Zernic, "Experimental results of running TCP/IP over ATM on NASA ACTS HDR", NASA/GRC, 1997, http://acts.grc.nasa.gov/library/docs/gsn/welchpaper.pdf

[7] Internet Engineering Task Force, "A Proposal to add Explicit Congestion Notification (ECN) to IP", RFC–2481, January 1999

[8] Internet Engineering Task Force "TCP Selective Acknowledgement Options", RFC–2018, April 1996

[9] G Morabito, I Akyildiz, S Palazzo, "TCP Peach: A Transport Layer Protocol for Satellite IP Networks", Second International Workshop on Mobile and Wireless Communication Networks, Paris France, May 2000

[10] H Price, J Ward, "Pacsat Broadcast Protocol", ARRL 9th Computer Networking Conference, pp. 232-238, August 1990.

[11] C Miller, "StarBurst MFTP Compared to Today's File Transfer Protocols: A White Paper", StarBurst Communications Corporation, 1996

[12] Consultative Committee for Space Data Systems, "CCSDS File Delivery Protocol (CFDP)--Part 1: Introduction and Overview", CCSDS 720.1-G-0.5, July 1999

[13] Internet Engineering Task Force, "NFS: Network File System Protocol Specification", RFC–1094, March 1989

[14] Internet Engineering Task Force, "The TFTP Protocol (Revision 2)", RFC–1350, July 1992

[15] Internet Engineering Task Force, "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC-1305, March 1992

[16] Internet Engineering Task Force, "File Transfer Protocol", RFC-959, October 1985

[17] Internet Engineering Task Force, "Hypertext Transfer Protocol -- HTTP/1.1", RFC–2616, June 1999

[18] Internet Engineering Task Force, "Simple Mail Transfer Protocol", RFC-821, August 1982

[19] Consultative Committee for Space Data Systems, "Space Communications Protocol Specification (SCPS) - RATIONALE, REQUIREMENTS, AND APPLICATION NOTES", CCSDS 710.0-G-0.3, April 1997

[20] "ITT Reference Data for Radio Engineers", Howard W. Sams & Co. Inc., ISBN 0-672-21218-8, 1975

[21] Internet Engineering Task Force, "IP Header Compression", RFC–2507, February 1999

[22] "Cellular IP - A New Approach to Internet Host Mobility," ACM Computer Communication Review, January 1999

[23] Internet Engineering Task Force, "IP Mobility Support", RFC-2002, October 1996

[24] J Rash, K Hogie, R Parise, E Criscuolo, F Hallihan, T Le, "Demonstrations of Internet Protocols in Space Using TDRSS", presented at The First Joint Space Internet Workshop, NASA Goddard Space Flight Center, November 2000

[25] J Rash, K Hogie, R Parise, E Criscuolo, J Langston, C Jackson, H Price, "Results of 'Internet in Space' Tests Using UoSat–12", presented at The First Joint Space Internet Workshop, NASA Goddard Space Flight Center, November 2000

[26] Internet Engineering Task Force, "Security Architecture for the Internet Protocol", RFC-2401, November 1998

[27] Internet Engineering Task Force, "Virtual Private Networks Identifier", RFC-2685, September 1999

Edward Criscuolo Jr. *joined Computer Sciences Corp. in 1991 as a Senior Computer Scientist working for the Goddard Space Flight Center. In that time, he has been the task lead for a number of spacecraft ground system projects that span many aspects of Goddard space missions, including Planning & Scheduling systems, spacecraft command management, and level-0 processing of telemetry and science data. In 1999, Mr. Criscuolo joined the OMNI project as a senior project member, where his duties include systems analysis, systems engineering, top-level design, prototype development, and backup technical lead.*

Keith Hogie *- Computer Sciences Corporation - Mr. Hogie has an extensive background in designing and building satellite data processing systems, control centers, and networks at GSFC. He has developed ground data processing systems and control centers for over 14 spacecraft over the last 25 years at NASA/GSFC, and led the development of the NASA Internetworking Laboratory Environment in 1990. He is the technical leader of the OMNI project at GSFC where he is applying his networking and satellite background to develop and demonstrate new communication technologies for future space missions.*

Dr. Ron Parise *- Computer Sciences Corporation - In 1984, Dr. Parise was selected as a payload specialist astronaut and was involved in mission planning, simulator development, integration and test activities, flight procedure development, and scientific data analysis. He has logged 615 hours in space as a member of the STS-35 and STS-67 crews. In 1996 Dr. Parise assumed a communications engineering support role for Mir, International Space Station (ISS), and the X-38 project. In 1997 Dr. Parise also began working with the OMNI project as a scientific liaison and systems architect.*